Information and Computer Science

Scruffy Text Understanding:
Design and Implementation of *Tolerant* Understanders

Richard H. Granger

Artificial Intelligence Project
Technical Report #186
February 1982

UNIVERSITY OF CALIFORNIA
IRVINE

Scruffy Text Understanding: Design and Implementation of *Tolerant* Understanders

Richard H. Granger

Artificial Intelligence Project
Technical Report #186
February 1982

This research was supported in part by the Naval Ocean Sytems Center under contract N00123-81-C-1078.

Scruffy Text Understanding: Design and Implementation of the NOMAD System

Richard H. Granger, Jr.
James R. Meehan
Chris Staros
Rika Yoshii

Artificial Intelligence Project
Department of Information and Computer Science
University of California
Irvine, California 92717

ABSTRACT

Most large text-understanding systems have been designed under the assumption that the input text will be in reasonably "neat" form, e.g., newspaper stories and other prepared texts, which typically consist of well-formed sentences and a logical order of presentation of concepts. However, many everyday uses of natural text (e.g., business memos, phone messages, notes) are very ill-formed, containing misspelled words, ungrammatical or only partially complete sentences, and poorly organized ideas.

Many large organizations employ humans whose sole job it is to take ill-formed messages and encode them into machine-readable form to be entered into a database. In such cases, there would be great potential benefit if that encoding process could be partially automated. This requires a "scruffy text understander"; i.e., a system that has the ability to correctly analyze the content of such texts in spite of their scruffiness.

This paper describes the design and implementation of the NOMAD system, which partially automates the encoding of poorly written Navy messages into well-formed formats. A number of problems are described which arise in the scruffy text domain that have not been dealt with in previous text-understanding systems.

This research was supported in part by the Naval Ocean Systems Center under contract N-00123-81-C-1078.

1.0 Introduction

Consider the following (scribbled) message, left by a computer science professor on a colleague's desk:

[1] Met w/chr agreed on changes to prposl nxt mtg 3 Feb.

A good deal of informal text such as everyday messages like the one above are very ill-formed grammatically and contain misspellings, ad hoc abbreviations and lack of important punctuation such as periods. Yet people seem to easily understand such messages, and in fact most people would probably understand the above message just as readily as they would a more "well-formed" version:

"I met with the chairman, and we agreed on what changes had to be made to the proposal. Our next meeting will be on Feb. 3."

No extra information seems to be conveyed by this longer version, and message-writers appear to take advantage of this fact by writing extremely terse messages such as [1], apparently counting on the reader's ability to analyze it in spite of its messiness.

As long as such messages are intended for a readership of one, there is no reason to worry about automatically processing such memos. However, in large organizations where informational messages are routinely passed through many hands, there is almost always a human who must intervene to put the message into more readable form before it is passed on. An extreme case of such an organization is the Navy, which every hour receives many thousands of short messages, each of which must be encoded into computer-readable form for entry into a database, and, when appropriate, should be routed on to specific readers for further evaluation. In such an organization, there is an obvious benefit to partially automating this encoding process. However, existing text-understanding systems (e.g. ELI [Riesbeck and Schank 76], SAM [Cullingford 77], FRUMP [DeJong 79], IPP [Lebowitz 80]) would fail to successfully analyze ill-formed texts like [1].

This paper investigates both the properties of scruffy texts like [1] that allow readers to understand it in spite of its scruffiness, and the knowledge and mechanisms that underlie readers' abilities to understand such texts. A text-processing system called NOMAD is discussed which makes use of the theories described here to process scruffy text in the domain of everyday Navy messages.

2.0 Background: Tolerant text processing

2.1 FOUL-UP: Figuring out unknown words from context

The FOUL-UP program [Granger 1977] was a component of the SAM system [Cullingford 77], which was a script-based story understander. FOUL-UP was used when SAM's conceptual analyzer (ELI) encountered an unknown word. FOUL-UP used the analyzer's own syntactic and semantic expectations (e.g., noun phrase, physical object) to create a temporary definition that would allow the analyzer to continue. When the parsed version of the sentence was being incorporated into the script by the Script Applier, FOUL-UP would revise the definition by combining the Applier's expectations about what role the unknown word played in the current script.

For example, FOUL-UP would have to intervene in the processing of the following text: "Friday, a car swerved off Route 69. The car struck an elm." The conceptual analyzer didn't know the word "elm", but it was expecting a noun phrase referring to either a physical object or a human, based on the sentence analyzed so far. The Script Applier expected that what the car struck fills the role of &OBSTRUCTION in the "vehicle accident" script. FOUL-UP matches these expectations with each other and creates a definition for "elm".

2.2 Blame Assignment in the NOMAD system

But even if SAM had known the word "elm", it would not have been able to handle a less edited version of the text, such as this: "Car swerved off Route 69 struck an elm." While human readers would have no great difficulty understanding this text, even if "elm" were an unknown word, no existing computer programs could do so. The difficulty here is syntactic, and it seems that a clause- or sentence-boundary has been omitted (e.g., perhaps the words "and it" were omitted).

These remarks reflect a three-step process in handling errors:

1. Notice error. Below, we show several ways in which this can happen. In this example, the verb "struck" occurs in a place where no verb is expected.

2. Figure out the source of the error ("Blame Assignment"). In this example, a verb has been encountered after a complete clause, i.e., where some kind of clause boundary probably should have been marked, say, by a period.

3. Correct the error, using a variety of heuristics, some of which are described below. In this example, we use the context of a car accident to figure out that the unspecified agent is probably the car, and we rephrase the text to allow that interpretation.

Sentences in Navy messages often have these kinds of problems. Consider for instance the following report of a simulated battle:

[2] LOCKED ON OPEN FIRED DESTROYED

By using a small amount of morphological knowledge, NOMAD is able to ignore the error in the past participle (i.e., it should be "opened fire"), and by detecting sentence-boundary conflicts as explained above, it is able to separate the message into three sentences, assuming that in each case the missing subject is the ship sending the message (and that there is an implicit target ship which has been attacked and destroyed).

Not all problems are syntactic, however, and a scruffy text understander must be able to correct semantic errors as well. An example combining both syntactic and semantic error correction is presented in the next section.

3.0 Operation of the NOMAD system

Consider the following messages:

[3] RETURNED BOMBS LEFT BY ENTERPRISE TO BASE
[4] RETURNED BOMBS LEFT ENEMY UNIT BURNING DEAD IN WATER

NOMAD has several word-senses for the word "left", including its definition as a past participle and as a past-tense verb. In example [3], it correctly assumes that "returned" is a verb meaning "took back to someone or someplace". Since there is no agent specified, it assumes that the sending ship is the actor, in this case, the Midway. "Bombs" is clearly the object of the verb. When it encounters the word "left", NOMAD assumes that the past participle is being used, since it is preceded by a noun phrase, and it constructs a representation for a relative clause in which someone as yet unspecified has left the bombs somewhere. "Enterprise" is then seen as the "leaver" of the bombs, and the base is their ultimate destination.

Example [4] again uses the words "returned" and "left", and NOMAD will of course make the same assumptions this time as last time about their usage. This time however both assumptions will turn out to be incorrect, and will have to be corrected. In particular, up till the word "enemy", the

two messages are identical. When NOMAD sees "enemy", however, it registers an error, since there is no expectation of an agent expressed directly as a noun phrase. [Step 1: an error has been noticed.] For syntactic reasons, therefore, it backs up and considers an alternative sense of the word "left", namely, a past-tense verb. Given that there is already a main verb, NOMAD assumes that there is a sentence-boundary problem [step 2: assigning the blame], and it splits the message into two sentences at this point [step 3: fix the error]. Since the new second sentence begins with a verb, it also assumes that the Midway is the actor. It goes on understand the rest of the sentence as the result of some battle, since an enemy has been injured.

At this point, however, there is another error encountered, this time a semantic one. The interpretation so far of the first part of [4] (PTRANSing some bombs somewhere) is anomalous in the context of a battle, which was just inferred. [step 1]. The word principally responsible for this interpretation is "returned". The inference module in NOMAD detects this error, assigns blame to the wrong choice of word-sense for "returned" [step 2], and directs the parser to re-interpret the first sentence ("RETURNED BOMBS") [step 3], which it does, using PROPEL as the basis for the representation.

After both syntactic and semantic backing-up, the message is completely parsed, and the resulting interpretation is sent to a natural language generator to provide a paraphrase of the original message.

4.0 Categories of tolerant understanding

4.1 Noticing errors

We have established at least four ways in which errors can be noticed. This is not meant to be an exhaustive list; it simply reflects the current state of the NOMAD system:

1. A word is unknown. This is easy to detect, and NOMAD immediately applies a spelling corrector in hopes of finding a word that it does understand.

2. The wording is clear, but it implies something that is contradictory, as in "Ship flying". We use an object-hierarchy to detect such cases, and a backup strategy is also useful here. For example, a message like "Unable to see color ship flying" is probably not two sentences, but rather refers to colored flags that the ship is displaying (flying).

3. The input message doesn't match the script for the current situation, either because the information is outside the script or because an essential, strongly predicted piece of information is missing.

4. The input message doesn't make sense in terms of the goals inferred by the system. "Returned bombs to base" implies transporting some weapons, while "Returning bombs to Kashin" is part of a battle. Only the recognition of goals allows us to distinguish between these two cases.

4.2 Assigning blame

We have also identified three sources for an error:

1. A word is unknown (and the spelling corrector fails). This is where the FOUL-UP component applies, figuring out the meaning of unknown words from context.

2. A word is known, but we have used the wrong sense of that word. If alternative meanings of the word are known to the system, we can back up and try them.

3. There is a syntactic problem, such as ambiguity ("Kashin attacked" can mean that it attacked us or we attacked them) or missing sentence boundaries ("Locked on open fired destroyed").

4.3 Correcting the problem

The general approach to error correction in NOMAD is to make use of any available contextual information, in the form of expectations at the word and concept levels, to help figure out how the message should have been interpreted. This approach gives rise to three specific categories of error correction so far employed by NOMAD:

1. "fast fixes" (e.g., an automatic spelling corrector), which do not do any deep analysis of the error, but rather attempt to find simple solutions to the problem in case one exists. In particular, all unknown words encountered by NOMAD are first put through a spelling corrector to match the word against known words, before calling in the FOUL-UP mechanism (see Section 2.1 above). If the spelling corrector fails, NOMAD then uses a simple morphological analyzer to strip off suffixes (e.g., -s, -ed, -ing), and looks to see whether the root is known, or whether a spelling correction can

apply to the root.

2. Choose alternate word sense. If blame for the error can be narrowed down to a particular word, then NOMAD checks to see whether that word has alternate word-senses that could apply in the surrounding context of this sentence. If so, that word sense is used and the rest of the sentence is re-analyzed in light of the expectations generated by the new word-sense (as in the example in Section 3).

3. Taking a different (syn-) tack. When blame has been assigned to a word that appears in a spot in the text at which its part of speech was not expected (as is "returned" in "Kashin taken under fire returned to base"), NOMAD assumes that some punctuation mark (usually a period) should have appeared just before the unexpected word. Then that portion of the message is re-analyzed with the assumed period in position.

REFERENCES

Richard E. Cullingford. Script Application: Computer Understanding of Newspaper Stories. PhD dissertation, Yale University, 1977. Research Report 116, Yale Computer Science Department.

Gerald DeJong. Skimming Stories in Real Time: An Experiment in Integrated Understanding. PhD dissertation, Yale University, 1979. Research Report 158, Yale Computer Science Department.

Richard H. Granger, Jr. FOUL-UP: A program that figures out meanings of words from context. In International Joint Conferences on Artificial Intelligence, Proceedings of the Fifth Conference, 1977, pages 172-178.

Michael Lebowitz. Integrated Partial Parsing. PhD dissertation, Yale University, 1980.

Christopher K. Riesbeck and Roger C. Schank. Comprehension by computer: expectation-based analysis of sentences in context. Research Report 78, Yale Computer Science Department, 1976.